SPECIFICATION

Electronic Version 1.2.8

Stylesheet Version 1.0

CAUSE AND EFFECT LOGIC APPLICATION IMPLEMENTATION

Cross Reference to Related Applications

This application claims the benefit of U.S. Provisional Application No. 60/201,001, filed May 1, 2000, which is hereby incorporated by reference in its entirety.

Background of Invention

[0001] This invention relates generally to cause and effect logic application implementation and, more specifically, to specifying functional requirements for cause and effect logic applications.

[0002] Some safety related applications utilize cause and effect specifications. However, no recognized international standards apply to definitions used in cause and effect logic. Various industries, companies and individuals adopt their own standards. Additionally, each cause and effect definition includes a corresponding set of cause and effect notations that typically can only express relatively simple combination logic. This inability to express more than simple combination coupled with the aforementioned lack of standards sometimes results in a safety system application logic definition which is ambiguous and hence subject to interpretation error.

Summary of Invention

[0003] In one embodiment, a method for Cause and Effect application logic implementation includes defining a formal methodology for specification of functional requirements for a target system based upon Cause and Effect notation and function blocks and employing a computer-aided specification tool-set to support capture and validation of functional requirements. The method further includes employing a

software module to directly execute Cause and Effect application logic.

[0004] In another embodiment, a computer for Cause and Effect application logic implementation is configured to receive a defined formal methodology for specification of functional requirements for a target system based upon Cause and Effect notation and function blocks. The computer is further configured to capture at least one functional requirement, validate at least one captured functional requirement, and directly execute Cause and Effect application logic.

[0005] In yet another embodiment, a database includes data relating to Cause and Effect notation and data relating to at least one Function block.

Brief Description of Drawings

[0006] Figure 1 is a block diagram of a system.

[0007] Figure 2 is a block diagram of an exemplary safety system.

[0008] Figure 3 is a flow chart for one embodiment of a method for Cause and Effect application logic implementation.

[0009] Figure 4 illustrates dividing functional requirements into a plurality of functional areas.

[0010] Figure 5 illustrates a Cause and Effect instruction set.

[0011] Figure 6 illustrates an example of Cause and Effect logic utilizing the Cause and Effect instruction set shown in Figure 3.

[0012] Figure 7 illustrates a generic example of Cause and Effect logic utilizing the Cause and Effect instruction set shown in Figure 3.

[0013] Figure 8 illustrates that a Cause and Effect gate logic is two-tiered.

[0014] Figure 9 is a Cause and Effect chart including a device control function block.

[0015] Figure 10 illustrates the relationship between a device control block and a Cause and Effect chart.

[0016] Figure 11 illustrates a Cause and Effect chart with an embedded Function Block.

[0017] Figure 12 illustrates the relationship between device control blocks and the Cause and Effect chart.

[0018] Figure 13 illustrates the additional relationship between device control blocks, the Cause and Effect chart, and an input conditioning block.

[0019] Figure 14 illustrates a device control function block with an associated HMI system graphical elements.

Detailed Description

[0020] There is herein provided a formal methodology for implementation of cause and effect application logic in a safety-related application for a programmable logic controller (PLC) system. It is contemplated that the benefits of the present invention accrue to all implementations of cause and effect application logic including implementations in non-safety related applications for systems other than PLC systems. As shown in Figure 1, the method utilizes at least one computer 10 in communication with at least one PLC 12 forming a system 14. As shown in Figure 2, an exemplary safety system 16 includes an HMI (Human Machine Interface) 18, a Fire and Gas protection system 20, a duplex Emergency Shutdown System 22 and a triplex Emergency Shutdown System 24. As used herein, the term computer is not limited to just those integrated circuits referred to in the art as computers, but broadly refers to computers, processors, microcontrollers, microcomputers, application specific integrated circuits, and other programmable circuits. In one embodiment, the PLCs are accessible to the computer via a network such as an intranet or the Internet. In an alternative embodiment, the PLC is directly connected to the computer. In another embodiment, the PLCs and the computers are interconnected to a network, such as a local area network (LAN) or a wide area network (WAN), through many interfaces including dial-in-connections, cable modems and high-speed ISDN lines.

[0021] Figure 3 is a flow chart for one embodiment of a method 30 for Cause and Effect application logic implementation. Method 30 includes defining 32 a formal methodology for specification of functional requirements for a target system based

upon Cause and Effect notation and function blocks. Method 30 further includes employing 34 a computer-aided specification tool-set to support capture and validation of functional requirements and employing 36 a software module to directly execute Cause and Effect application logic.

[0022] As explained in greater detail below, a user defines 32 a formal specification for a particular application and specifies at least one requirement according to the defined formal methodology. The user employs 34 a computer-aided specification tool-set which captures and validates the specified requirements. The user also employs a software module which directly executes the specified requirements. In one embodiment, the software module is an execution engine that is loaded on the target system such as a target PLC system and the engine directly executes the specified requirements without translation. When a user desires to utilize an existing collection of specified requirements on a different target system, the user ports the engine to the new device and the application specification (i.e. collection of specified requirements) remains the same. Accordingly, the different target system is a new device or an upgraded platform for an existing device but the application specification need not be changed. Since, the application specification need not be changed, the new or upgraded platform is functional without the costs associated with changing the application specification.

[0023] As illustrated in Figure 4, method 30 includes dividing functional requirements into a plurality of functional areas 38 including a device control area 40 and a plant control area 42. By specifying control requirements of these different functional areas 40 and 42, an overall methodology is simplified and a plant control specification is kept simple. Additionally, dividing the functional requirements facilitates results that closely resemble American Petroleum Institute standards such as API 14C Safe Chart definitions, which have been used by safety engineers for many years.

[0024]

After dividing functional requirements of the target PLC system into functional areas 38, method 30 further includes defining logic requirements for both device control 40 and function control 42. In one embodiment, the logic requirements for device control 40 include controlling and/or monitoring a particular type of field

device 44 in the target PLC system and the requirements are defined by a re-usable device control function block (not shown in Figure 4). When a device control function block is defined for a specific field device, then that particular block is used throughout a safety system specification wherever another of that specific device needs to be controlled. Functional areas 38 are associated with at least one operational requirement 46 and at least one maintenance requirement 48 which are initially addressed during defining 12 formal methodology for a target system 50.

[0025]

Figure 5 illustrates a Cause and Effect instruction set 52 including a plurality of logic symbols 54 utilized by an user to specify Boolean logic requirements. Logic symbols include an OR symbol (X_n) 56, an INV OR symbol

($\overline{X}_n$) 58, an AND symbol ($\&_n$) 60, and an INV AND symbol ($\overline{\&}_n$) 62. Logic symbols 54 further include an ENABLE symbol (E) 64, an INV ENABLE symbol ($\overline{E}_n$) 66, an ONTIMER symbol ($T(NN)_n$) 68, an INV ONTIMER symbol($\overline{T}(NN)_n$) 70, a RESET symbol (R) 72, and an INV RESET symbol ($\overline{R}$) 74.

[0026] As explained below, Cause and Effect instruction set 52 is utilized in Cause and Effect charts together with Cause and Effect control function blocks (not shown in

Figure 3). OR symbol 56 indicates that an input term is or'ed into a gate referenced by the n subscript of X_n. INV OR 58 indicates that the input term is inverted and or'ed into the gate. AND 60 indicates that the input term is and'ed into the gate. INV AND 62 indicates that the input term is inverted and and'ed into the gate. ENABLE 64 indicates that the input term is or'ed with other ENABLEs 64. INV ENABLE 66 indicates that the input term is inverted and or'ed with other ENABLEs. ENABLE 64 and INV ENABLE 66 are utilized to enable or'ing (OR 56 and INV OR 58) and and'ing (AND 60 and INV AND 62) for the gate. If ENABLE 64 and INV ENABLE 66 are not defined then the gate is enabled by default. ONTIMER 68 indicates that the input term is subject to an on delay of NN seconds and after NN seconds the input term is or'ed into a group referenced by the n subscript of T(NN)n. INV ONTIMER 70 indicates that the input term is inverted and subject to an on delay of NN seconds and after NN seconds the inverted term is or'ed into the group. RESET 72 indicates that the input term resets a latch. INV RESET 74 indicates that the input term resets a latch when the input term is false. A latch set term has priority and if no reset terms are defined for a gate then that gate is non-latching. Each symbol 54 includes a subscript (n) when referring to a specific gate and does not have a subscript when referring to a base gate (not shown in Figure 5). Instruction set 52 contains only five separate Boolean instructions (OR 56, AND 60, ENABLE 64, ONTIMER 68, and RESET 72) along with the negative or inverse of those five (INV OR 58, INV AND 62, INV ENABLE 66, INV ONTIMER 70, and INV RESET 74). In alternative embodiments, instruction set 52 includes more than and less than five separate instructions along with their inverses but always contains less than ten separate instructions to impart a limited ability of Cause and Effect notation (instruction set 52) to express anything more complex than simple combinational logic. This limited ability allows for a notation that is easily interpreted by both process/safety engineers and system builders.

[0027] Figures 6 and 7 illustrate examples of Cause and Effect logic utilizing instruction set 52 with a base gate 76 and an input gate 78 including at least one output 80. Logic execution proceeds with outputs 80 from gate 78 feeding in as OR terms into base gate 76 to be solved along with other base gate input terms which include OR terms 82, TIMER terms 84, AND terms 86, ENABLE terms 88, and RESET terms 90.

Additionally, as shown in Figure 6, some AND terms 86 (shown with an &) are sent to base gate 76 and another AND term 86 (shown with an $\&_1$) is sent to input gate 78. Individual gates may or may not include all input terms noted in Figure 6. Where input terms of a given type are not defined then an associated logic is defeated in such a manner as to allow normal gate function e.g. if no reset terms are defined then the gate is non-latching.

[0028] Furthermore, as depicted by Figure 8, the Cause and Effect gate logic is two-tiered in that a plurality of Cause and Effect Gate inputs 92 are provided wherein some of inputs 92 are directly received by a Cause and Effect Base Gate 94 while other inputs 92 are received by a plurality of Cause and Effect Input Gates 96 whose outputs are received by Cause and Effect Base Gate 94. As explained above, the logic operators of Cause and Effect Base Gate 94 have no suffix whereas Cause and Effect Input Gates 96 are denoted by a suffix to the logic term.

[0029] Figure 9 is a Cause and Effect chart 98 including a device control function block 100 which is sometimes referred to as an IAM block for Intelligent Actuation and Measurement. Function block 100 is a complete logic element and includes target system executable logic tags including input signal tags representing alarm states and output signal tags representing shutdown actions for the target PLC system. Additionally function block 100 include Human Machine Interface (HMI) data, operation data, and maintenance data.

[0030] Figure 10 illustrates the relationship between device control block 100 and Cause and Effect chart 98 shown in Figure 8. Inputs to function block 100 are set by outputs from Cause and Effect chart 98. For example, a demand result (ZD) 102 for function block 100 is driven by logic within Cause and Effect chart 98. Additionally, outputs of function block 100 are available as inputs to Cause and Effect chart 98. For example, an Out1 104 for function block 100 produces an alarm signal FLT 106 indicating that a valve (not shown) has not moved into a demanded position. Similarly, a pressure transmitter 108 produces a pressure high alarm that is used to control logic within Cause and Effect chart 98.

[0031] Figure 11 illustrates a Cause and Effect chart 112 with an embedded control

function block 110 exemplifying control of a blowdown valve (not shown) that supports manual operation 114, automatic operation 116, and a manual blowdown abort facility 118. Control function block 110 is created with a plurality of inputs and outputs and then placed on Cause and Effect chart 112 and connected to a plurality of input and output tags 120.

[0032] Figure 12 illustrates the relationship between device control blocks and a Cause and Effect chart 134 Method 30 (shown in Figure 1) allows a straightforward connection of alarm states from a first Pressure Transmitter IAM 130 to output demands for safety system actuators from a Valve Controller IAM 132 using Cause and Effect chart 134. Figure 13 illustrates the additional relationship between device control blocks, Cause and Effect chart 134, and an input conditioning block 140. Additionally, Figure 13 demonstrates how adding a second and third Pressure Transmitter IAMs 136 and 138 does not obscure logic specification and implementation. Pressure Transmitter IAMs 130, 136, and 138 are connected to Cause and Effect chart 134 via Input Conditioning Voter Block 140 such as a 2oo3 Voting block. Input Conditioning Voter Block 140 is a function block that derives consolidated and/or modified signals from one or more physical or logical inputs. Input Conditioning Voter Block 140 is placed in a separate column in Cause and Effect chart 134 with logical connections to a plurality of pressure transmitter IAM function blocks (not shown). Accordingly, Pressure Transmitter IAM 130, 136, and 138 are all identical and are each associated with identical pressure transmitter IAM function blocks which are connected to Input Conditioning Voter Block 140 in Cause and Effect chart 134 One embodiment of method 30 utilizes an application software package to capture and validate functional requirements for a target PLC system. The application software is capable of developing a database of logic symbols and function blocks (including Cause and Effect instruction set 52 and at least one of a Cause and Effect control function block, a device control function block, and an input conditioning function blocks that are directly implemented using executive software.

[0033] The database is initially formatted as a standard database utilizing Cause and Effect instruction set 52 and function blocks with pre-defined executable logic in a manner that is independent of the target PLC system. This standard database is

focused on ensuring that the specification is correct and unambiguous, not on the target PLC language set. Since most current PLC platforms support a very similar language set, the effort to implement the function blocks on any specific target PLC is not prohibitive.

[0034] The applications software also provides for data import or an on-line definition of field devices. Additionally, the applications software provides for creating new function block templates and user-defined function blocks may be created in any language supported by the target PLC system, for example, ladder diagram, function block diagram (block logic) and Sequential Function Chart (sequences depicted graphically as steps and transitions).

[0035] Figure 14 illustrates a device control function block 152 with associated HMI system graphical elements 150. An HMI database is automatically populated when the device control function block is created. This population is performed directly on-line if supported by a selected Supervisory Control and Data Acquisition (SCADA) system or a distributed control system (DCS) system or via a generic file import system. Graphical icons and pop-ups associated with the device control function block are automatically generated with certain Windows NT based HMI systems that support Active X technology. Device signal input and output tags are also automatically generated when the function block is created. This use of function block logic significantly reduces error since there is no need to manually link display points to database tags.

[0036] When specific logic signals are required in support of an HMI interface (such as Any Alarm in a Unit) then these specified logic signals are generated on Cause and Effect charts dedicated to the HMI interface. HMI requirements include a graphical icon representing the device or control function, an operator control pop-up, a maintenance control pop-up, and an alarm priority and data logging information on a function block signal basis.

[0037] The application software is also used to construct a Cause and Effect specification chart from functional requirements of the target PLC system. The Cause and Effect charts utilize a conventional layout for inputs and outputs. To support large designs,

each chart includes one or more sheets. Additionally, points that are outputs of charts are used as inputs to the same chart or a different chart. In this way, inter-trips between Unit Shutdown charts or connections to and from Master Shutdown logic are generated.

[0038] Dimensions and layout of the specification chart is generally defined once for a project and is then used for all charts. Each column on a Cause and Effect logic chart is used to represent a multi-variable input single output Cause and Effect function. The Cause and Effect charts are constructed by selecting a desired device control block signal tag (e.g. 01-PAHH-1234), from a database held within the applications software program, and placing it on an input and/or an output function chart section as appropriate. Signal identification fields are automatically expanded to define the IAM function block reference (e.g. AI1 for Analogue Input type 1) and a field device tag (01-PT-1234) associated with a cause tag.

[0039] The Cause and Effect notation are selected from the applications software database and placed on the chart at an intersection of input signal tags and output signal tags corresponding to a direct connection of alarm states to shutdown actions for the target PLC system. A plurality of Cause and Effect control function blocks are selected from the database and placed on the chart with defined input and output pins of those function blocks connected to input and output signal tags respectively from device control function blocks. HMI interface logic is placed on the Cause and Effect chart and input conditioning function blocks are placed in a separate column on the chart. To enhance integrity, it is not possible to place any input or output signal tag, function block, or symbol on a chart unless it is already defined in the applications software database.

[0040]

A formal methodology to capture requirements of a safety system"s functionality in terms of function block interconnections and Cause and Effect logic notation is provided. Once captured, these requirement specifications are subjected to pre-defined validation rules implemented through the application software. Validation rules include conformance of Cause and Effect notation to specified structure, validity of function block connections and data types, and consistency of defined functional

requirements and safety system input/output index.

[0041] The application software is also used in modifying a Cause and Effect specification chart post implementation. Any application logic modification required is performed directly on the Cause and Effect chart specifications, which is subsequently be loaded on-line to the target PLC system. Accordingly, documentation and implementation are always equal. The Cause and Effect logic captured on a Cause and Effect chart is directly executed by an executive software package loaded in a target PLC system supported by a database customized and re-validated to the target PLC system. Therefore, no user translation of Cause and Effect logic to the target PLC system is required.

[0042] While the invention has been described in terms of various specific embodiments, those skilled in the art will recognize that the invention can be practiced with modification within the spirit and scope of the claims.